情報処理学会論文誌　Vol. 51　No. 12　2216–2226 (Dec. 2010)

SMT/CMP 向け固定優先度スケジューリング用動的電圧周波数制御の提案と RMT Processor を用いた実機評価

池田雄児[†1]　加藤真平[†1]　山崎信行[†1]

組み込みシステムにとって省電力化は重要な課題の 1 つであり，プロセッサの動作電圧および動作周波数を動的に制御することで消費電力量の削減を実現する動的電圧周波数制御が注目されている．また，性能に対する電力の効率が良いマルチプロセッサ技術の利用も注目されている．本論文では，固定優先度マルチプロセッサスケジューリングを対象として，プロセッサの消費電力量を効果的に削減する動的電圧周波数制御アルゴリズムを提案する．具体的には，タスクごとに設定するウィンドウ内においてプロセッサのアイドル時間を先読みし，そのアイドル時間からすべてのタスクのデッドラインを守ることが可能な最低周波数を算出する．電圧と周波数の制御が可能なリアルタイム処理用の SMT プロセッサである RMT Processor を用いて評価を行った結果，提案手法は静的に動作周波数を決定する手法よりも消費電力量を削減することができた．

Dynamic Voltage and Frequency Scaling for Fixed-priority Scheduling on SMT/CMP and Experiments on an RMT Processor

Yuji Ikeda,[†1] Shinpei Kato[†1]
and Nobuyuki Yamasaki[†1]

Dynamic voltage and frequency scaling (DVFS), which can reduce power consumption by scaling voltage and frequence of processors, has received attension, since saving energy is one of primary subjects for embedded systems. Also, a multiprocessor architecture has received attension, since it is more energy-efficient than a comparable uniprocessor architecture. In this paper, we present a DVFS algorithm that reduces power consumption for fixed-priority multiprocessor scheduling. Our algorithm computes the minimum frequency, at which all tasks are guaranteed to meet their deadlines, by looking ahead the idle time within a window assigned to every task. According to the results of experiments on an RMT Processor, which is SMT architecture for real-time system and on which DVFS is available, our algorithm reduced power consumption, as compared to the static voltage and frequency scaling algorithm.

1. はじめに

リアルタイムスケジューリングは動的優先度スケジューリングと固定優先度スケジューリングの 2 つに大別される[1)]．動的優先度スケジューリングは高いスケジュール可能性を実現できるが，システムの負荷が高いときに，デッドラインミスを起こすタスクが予測できないという欠点がある．一方，固定優先度スケジューリングはスケジュール可能性は動的優先度スケジューリングよりも低くなる場合が多いが，高い予測性や小さいジッタ，実装が容易であるという利点がある．このような利点から，固定優先度スケジューリングは実際のシステムにおいてしばしば利用される．

組み込みシステムでは電力は限られた資源であり，消費電力を抑えることは非常に重要である．動的電圧周波数制御はシステムが要求する性能を満たす範囲でプロセッサの動作周波数および動作電圧を動的に制御することにより，プロセッサの消費電力を削減する技術である[2)]．多くのシステムにおいて最も消費電力の大きな要素はプロセッサであり，プロセッサの消費電力はシステム全体の消費電力に大きな影響を与える．今日主流である CMOS 回路で構成されるプロセッサでは，c をスイッチング容量，V_{dd} を動作電圧とすると，演算ごとの消費電力は $E^{op} = cV_{dd}^2$ で与えられる[2)]．c が一定であると仮定すると，消費電力を削減するためには動作電圧を下げなければならない．f_{max} をプロセッサの最大動作周波数，V_t を閾値電圧とすると，最大動作周波数と動作電圧には $f_{max} \propto (V_{dd} - V_t)^2/V_{dd}$ という関係があり，動作周波数を下げることで動作電圧を下げることができる．つまり，動作周波数を下げることによって，消費電力量を削減することができる．

パフォーマンスを維持しながら消費電力を下げるもう 1 つの方法として，マルチプロセッサや同時細粒度マルチスレッディング（SMT）[3)] およびチップマルチプロセッシング（CMP）[4)] などの高並列アーキテクチャを用いることがあげられる．

リアルタイムスケジューリングに，プロセッサの電圧と周波数の制御を行い消費電力量を

†1 慶應義塾大学
Keio University

抑えるという概念を加えたものがリアルタイム電圧周波数制御である．これまでに，シングルプロセッサ向けの動的電圧周波数手法が数多く提案されている[5)–7)]．また，マルチプロセッサ向けの手法も提案されている[8)–11)]．これらのマルチプロセッサ向けの手法ではプロセッサごとに動作周波数および動作電圧を独立して設定できると仮定している．しかしながら，SMT や CMP のシステムにおいてはこの仮定は適さない．SMT ではプロセッサの動作周波数を変化させるとすべてのスレッドに影響してしまう．また，CMP では，コアごとに独立して動作周波数を設定することはできるかもしれないが，コアごとに異なる電圧を設定することは困難である．すべてのプロセッシングユニットが同じ電圧で動作する SMT や CMP のアーキテクチャを想定した手法はほとんど提案されていない．また，提案された手法を実機で評価した論文もほとんど存在しない．

本研究では，SMT や CMP で構成されたシステムにおいて，高い予測性や小さいジッタという長所を持つ固定優先度スケジューリング用のリアルタイム動的電圧周波数制御アルゴリズムを提案する．動作電圧と動作周波数の制御が可能なリアルタイム処理用の SMT プロセッサである RMT Processor を用いて評価を行い，提案手法の有効性を評価する．

本論文の構成は以下のとおりである．2 章では，本研究の関連研究について言及する．3 章では，本研究が対象とするシステムモデルを説明する．4 章では Rate Monotonic（RM）[1)] に基づいた SMT および CMP 向けのリアルタイム電圧周波数制御手法を提案する．5 章では Responsive Multithreaded Processor [12)] を使って提案手法の評価を行う．最後に，6 章で本研究の結論と今後の課題を述べる．

2. 関 連 研 究

シングルプロセッサ向けの手法には以下のようなものがある．Pillai ら[5)] は，動的優先度スケジューリングである Earliest Deadline First（EDF）[1)] 向けの Cycle-Conserving と Look-Ahead アルゴリズムを提案している．池田ら[6)] は，Look-Ahead アルゴリズムを固定優先度スケジューリング向けに拡張した手法 Look-Ahead Window アルゴリズムを提案している．Chen ら[7)] は，リーク電流を考慮したスケジューリング手法である P-Procrastination を提案している．

プロセッサごとに独立して動作電圧を設定するマルチプロセッサ向けの手法には以下のようなものがある．Chen ら[8)] は，フレームベースのタスクセットを対象とした消費電力量を最小化する近似スケジューリングアルゴリズムを提案している．フレームベースのタスクセットとは，周期およびデッドラインをすべて同じタスクで構成したものである．また Chen ら[9)] は，周期タスクを対象としリーク電流を考慮したスケジューリングを提案している．Shao ら[10)] は，アプリケーションのループ処理を利用したスケジューリング手法である DVLS を提案している．Zhu ら[11)] は，フレームベースのタスクセットを対象として，タスク間に依存性がある場合とない場合の両方のスケジューリング手法を提案している．

すべてのプロセッサが同じ動作電圧で動作する SMT や CMP を対象とした研究には以下のようなものがある．Yang ら[13)] は，フレームベースのタスクセットを対象として消費電力量を最小化する近似スケジューリングアルゴリズムを提案している．Neils ら[14)] は，グローバル EDF [15)] 向けの手法 MOTE を提案している．彼らはシステムの開始前に静的に動作周波数を決定する手法と，システムの実行時に最悪実行時間よりも早く実行を終了したときの余裕時間を利用し動作周波数を下げる手法を提案している．船岡ら[16)] は，マルチプロセッサ上でシステム使用率を 100%利用可能なスケジューリング手法と，そのスケジューリング向けの動的電圧周波数制御手法を提案している．これらの手法は，タスクの実行が動的優先度スケジューリングによってスケジューリングされている．そのためジッタが大きくなるという問題があり，ジッタが小さい必要がある制御系のアプリケーションには不向きである．

以上から，本論文では Look-Ahead Window [6)] を SMT/CMP 向けに拡張し，固定優先度スケジューリング向け動的電圧周波数制御手法を提案する．

3. システムモデル

システムは，m 個のスレッドまたはコアから構成される．これらのスレッドやコアを便宜上 LP（Logical Processor）と呼ぶ．m 番目の LP を LP_m と表す．すべての LP は同じ動作周波数および動作電圧で動作する．制御可能な周波数の集合を $f = \{f_1, \ldots, f_l | f_1 < \cdots < f_l\}$ とする．最高動作周波数に対する周波数の比を α（$0 \leq \alpha \leq 1$）とする．周波数制御手法が周波数の比 α を決定したとき，$\alpha \leq f_i/f_l$ を満たす最小の周波数 f_i をシステムに設定する．システムには，n 個の周期タスク $\tau = \{\tau_1, \ldots, \tau_n\}$ が存在する．C_i を最悪実行時間，P_i を周期，i 番目のタスクを $\tau_i(C_i, P_i)$ と表記する．それぞれのタスクは独立しており，共有リソースは考慮しない．また，それぞれのタスクはつねにプリエンプション可能である．システム実行中に新しいタスクが到着することはない．タスクは固定優先度スケジューリングである Rate Monotonic（RM）[1)] によってスケジューリングされる．RM は，各タスクに対して周期の短い順に高い優先度を割り当てる．$P_1 < \ldots < P_n$ とし，したがってタスク τ_i の優先度は i に反比例する．タスク τ_i の相対デッドラインは周期 P_i と等しく，このデッドラインまでに実行を終えなければならない．それぞれの LP で実行中のタスクをカレント

タスクと呼ぶ．また，ある時刻においてタスクの実行が終了していないとき，そのタスクはアクティブであると呼ぶ．タスク τ_i がアクティブであるとき，タスク τ_i の残り実行時間を R_i，デッドラインを d_i と表す．区間 $[t, t+W]$ におけるタスク τ_i のプロセッサ利用時間を $u_i(t, W)$ と定義する．区間 $(t, t+W]$ でタスク τ_i がリリースされる回数を k とすると，$u_i(t, W) = R_i + kC_i$ となる．

4. 動的電圧周波数制御アルゴリズム

本章では，シングルプロセッサ向けの DVFS 手法である Look Ahead Window（LAW）[6] を SMT および CMP 向けに拡張した手法を提案する．LAW は，タスクごとに設定するウィンドウ内においてプロセッサのアイドル時間を先読みし，そのアイドル時間からすべてのタスクのデッドラインを守ることが可能な最低周波数を算出する．プロセッサがアイドルする時間を余裕時間，現在時刻から各タスクのデッドラインまでの時間をウィンドウと呼ぶ．ウィンドウはタスクごとに異なる．すべてのタスクにおいてウィンドウ内の余裕時間を求め，その中の最小値を使って動作周波数を決定する．余裕時間の最小値を用いることで，すべてのタスクのデッドラインを守ることができるようにする．タスクの終了時およびタイマ割込みの周期ごとに，本章で提案するアルゴリズムで動作周波数と動作電圧を設定する．

LAW では，算出した余裕時間はすべてカレントタスクのみに利用し動作周波数を低減させる．最悪実行時間と実際の実行時間が等しい場合，算出した余裕時間は実行可能な全タスクで分け合う手法の方が，提案手法に比べ電力効率が良くなると考えられる．しかしながら，実際のシステムでは，実際の実行時間と最悪実行時間との差は大きい傾向にある．ある時刻で予測された余裕時間よりも，実際の余裕時間は大きくなるため，余裕時間をカレントタスクのみに割り当てても十分に消費電力量を削減できると考えられる．5.3 節で，それぞれの手法における消費電力量の比較結果を示す．

マルチプロセッサ用スケジューリングは，タスクをプロセッサに割り当てる方式から，パーティショニング方式とグローバルスケジューリング方式の 2 つに大別できる．パーティショニング方式は，あらかじめ静的に各プロセッサにタスクを割り当て，各プロセッサで独立にスケジュールを行う方式である．一方，グローバルスケジューリング方式は，タスクをシステム全体で 1 つのレディキューに入れ，高優先度のタスクから動的に各プロセッサにスケジュールする方式である．4.1 節ではパーティショニング方式の RM スケジューリング向けの手法を，4.2 節ではグローバル方式の RM スケジューリング向けの手法を提案する．

Algorithm: Look-Ahead Window on Partition RM

```
 1: foreach 1...m as i
 2:     foreach τ_j ∈ AS(i)
 3:         if τ_j is active then
 4:             W_j ← d_j − t_now
 5:             s_j ← W_j − Rj − Σ_{k∈HP(i,j)} u_k(t_now, W_k)
 6:         else
 7:             s_j ← P_n
 8:         end if
 9:     end foreach
10:     S ← min(s_j | τ_j ∈ AS(i))
11:     α_i ← R^i_ct/(S + R^i_ct)
12: end foreach
13: α ← max(α_1, ..., α_m)
```

図 1　LAW-PRM アルゴリズム
Fig. 1　LAW-PRM argorithm.

4.1 LAW on Partition-RM

LAW on Partition-RM（LAW-PRM）では，まずそれぞれの LP ごとに動作周波数を計算する．そしてすべての LP の動作周波数の中で最大のものをシステムの動作周波数として設定する．

図 1 に LAW-PRM のアルゴリズムを示す．現在時刻を t_{now}，タスク τ_j の余裕時間を s_j，ウィンドウを W_j，LP_i のカレントタスクの残り実行時間を R^i_{ct} と表す．また，LP_i に割り当てられているタスクの集合を $AS(i)$，LP_i に割り当てられているタスクのうちタスク τ_j よりも優先度が高いタスクの集合を $HP(i, j)$ と定義する．m は LP の個数である．まずタスク τ_j がアクティブの場合，ウィンドウ W_i を計算する（4 行目）．ウィンドウ内において，タスク τ_j と同じ LP に割り当てられ，かつ τ_j よりも優先度が高いタスクのプロセッサ利用時間を計算し，タスク τ_j の余裕時間 s_j を求める（5 行目）．タスク τ_j がアクティブではない場合は，余裕時間を計算する必要はない．ここでは便宜上，余裕時間の最大値である P_n を代入している（7 行目）．LP_i に割り当てられているタスクの余裕時間の最小値を求める（10 行目）．残り実行時間と余裕時間の最小値から LP_i の動作周波数を計算する（11 行目）．すべての LP の動作周波数の最大値を求める（13 行目）．

LAW-PRM アルゴリズムにおける動作周波数の算出例を**図 2** に示す．$t_{now} = 0$ の場合の動作周波数の比 α を求める．システムには 2 つの LP と $\tau_1(1, 5)$，$\tau_2(2, 6)$，$\tau_3(3, 8)$，$\tau(4, 11)$

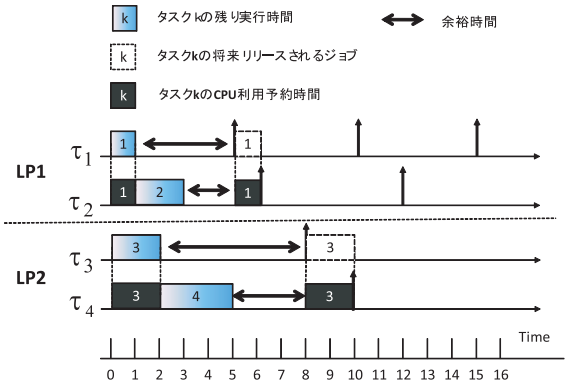


図 2　LAW-PRM アルゴリズムの動作例
Fig. 2　Example of LAW-PRM algorithm.

Algorithm: Look-Ahead Window on Global RM

```
1: foreach 1...n as i
2:     if τ_i is active then
3:         W_i ← d_i − t_now
4:         s_i ← W_i − R_i − Σ_{j=1}^{i−1} u_j(t_now, W_i) / m
5:     else
6:         s_i ← P_n
7:     end if
8: end foreach
9: S ← min(s_1, ..., s_n)
10: α ← R_min/(S + R_min)
```

図 3　LAW-GRM アルゴリズム
Fig. 3　LAW-GRM argorithm.

の 4 つのタスクが存在するとする．τ_1 と τ_2 は LP1 に，τ_3 と τ_4 は LP2 に割り当てられているとする．LP1 のカレントタスクは τ_1，LP2 のカレントタスクは τ_3 である．また，残り実行時間は $R_1 = 1$，$R_2 = 2$，$R_3 = 2$，$R_4 = 3$ となっている．まず LP1 の動作周波数を計算する．タスク τ_1 の次の（今の場合は最初の）デッドラインは時刻 5 であるので，ウィンドウ $W_1 = 5 - 0 = 5$ となる．タスク τ_1 は高優先度タスクにブロックされることはないので，余裕時間は $s_1 = 5 - 1 = 4$ となる．タスク τ_2 において，次のデッドラインは時刻 6 であるので $W_2 = 6 - 0 = 6$ となる．区間 $[0, 1]$ および $[5, 6]$ はタスク τ_1 にブロックされると予想されるため，タスク τ_2 の余裕時間は $s_2 = 6 - 2 - 2 = 2$ となる．LP1 では，タスク τ_2 の余裕時間が最小なので，$\alpha_1 = 1/(2 + 1) = 0.33$ となる．次に LP2 の動作周波数を計算する．タスク τ_3 において，デッドラインは時刻 8 であるので，ウィンドウ $W_3 = 8$ となる．タスク τ_3 は高優先度タスクにブロックされることはないので，余裕時間は $s_1 = 8 - 2 = 6$ となる．タスク τ_4 において，次のデッドラインは時刻 10 であるので $W_2 = 10 - 0 = 10$ となる．区間 $[0, 2]$ および $[8, 10]$ はタスク τ_3 にブロックされると予想されるため，タスク τ_4 の余裕時間は $s_2 = 10 - 3 - 4 = 3$ となる．LP2 においては，タスク τ_4 の余裕時間が最小なので，$\alpha_2 = 2/(3 + 2) = 0.4$ となる．以上から，$t_{now} = 0$ における動作周波数の比は $\alpha = max(0.33, 0.4) = 0.4$ となる．

4.2　LAW on Global-RM

グローバルスケジュール方式では，スケジューラはジョブを実行する LP を実行時に動的に決定する．LAW on Global-RM（LAW-GRM）では，優先度が高いタスクの総実行時間を LP の数で割ることで，タスクが高優先度タスクにブロックされる時間を見積もり，余裕時間を計算する．

図 3 に LAW-GRM のアルゴリズムを示す．現在時刻を t_{now}，タスク τ_i の余裕時間を s_i，ウィンドウを W_i，カレントタスクの残り実行時間のうち最小のものを R_{min} と表す．m は LP の個数である．まずタスク τ_i がアクティブの場合，ウィンドウ W_i を計算する（3 行目）．タスク τ_i より優先度が高いタスクの総実行時間を LP の数で割ることで，タスク τ_i が高優先度タスクにブロックされる時間を見積もり，タスク τ_i の余裕時間 s_i を計算する（4 行目）．タスク τ_i がアクティブではない場合は，余裕時間を計算する必要はない．ここでは便宜上，余裕時間の最大値である P_n を代入している（6 行目）．すべてのタスクの余裕時間の最小値を求める（9 行目）．求めた余裕時間の最小値とカレントタスクの残り実行時間の最小値を用いて動作周波数の比 α を計算する（10 行目）．

LAW-GRM アルゴリズムにおける動作周波数の算出例を**図 4** に示す．$t_{now} = 0$ の場合の動作周波数の比 α を求める．システムには 2 つの LP と，$\tau_1(2, 8)$，$\tau_2(4, 9)$，$\tau_3(2, 11)$ の 3 つのタスクが存在するとする．残り実行時間は $R_1 = 2$，$R_2 = 4$，$R_3 = 2$ となっている．カレントタスクはタスク τ_1 と τ_2 である．タスク τ_1 において，次のデッドラインは時刻 8 であるので，ウィンドウ $W_1 = 8$ となる．タスク τ_1 は高優先度タスクにブロックされることはないので，余裕時間は $s_1 = 8 - 2 = 6$ となる．タスク τ_2 において，次のデッドラインは時刻 9 であるので $W_2 = 9 - 0 = 9$ となる．区間 $[0, 1]$ および $[8, 9]$ はタスク τ_1 にブロック

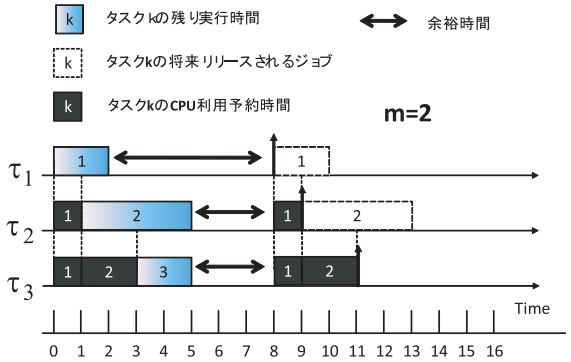


図 4　LAW-GRM アルゴリズムの動作例
Fig. 4　Example of LAW-GRM algorithm.

されると予想されるため，タスク τ_2 の余裕時間は $s_2 = 9-4-2 = 3$ となる．タスク τ_3 において，次のデッドラインは時刻 11 であるので $W_2 = 11-0 = 9$ となる．区間 $[0,3]$ および $[8,11]$ はタスク τ_1 およびタスク τ_2 にブロックされると予想されるため，タスク τ_3 の余裕時間は $s_3 = 11-2-6 = 3$ となる．以上から，余裕時間の最小値は $S = min(6,3,3) = 3$ となる．カレントタスクの残り実行時間の最小値は $R_{min} = R_1 = 2$ なので，動作周波数の比は $\alpha = 2/(3+2) = 0.4$ となる．

5. 評　　価

本章では，電圧周波数制御が可能なプロセッサを用いて提案手法の評価を行う．まず，5.1 節では評価環境について述べる．5.2 節では提案手法を静的な手法と比較する．5.3 節では，余裕時間を全タスクに平均的に分配する手法と，カレントタスクのみに分配する本論文の手法とを比較する．最後に，5.4 節でスケジューラの実行時間を評価する．

5.1 評 価 環 境

評価はリアルタイム処理用の SMT プロセッサである Responsive Multithreaded Processor（RMTP）[12] の評価ボードを用いて行う．RMTP は，リアルタイム処理用の SMT プロセッサであり，8 スレッド分のコンテキストと 4 つの命令パイプラインを用いた様々な実行モードを持っている．通常の SMT 実行を行う SMT モード以外にリアルタイム処理用実行モードとして大きく RMT 実行モード（優先度が高い順に並列実行する優先度付き SMT）と固定発行モード（Fixed Issue Mode）がある．固定発行モードはスレッドを命令パイプラインに割り当てる資源予約を行う実行モードである．本評価では 1 スレッドを 1 パイプラインに割り当てるように設定して固定発行モードを使用する．この設定により RMTP はマルチコアプロセッサのように振る舞う．RMTP の動作周波数と動作電圧は OS から動的に設定可能である．RMTP では内蔵のクロックジェネレータにより設計上は 2^{15} 種類の動作周波数が選択可能であるが，本論文では 4 種類に制限した．

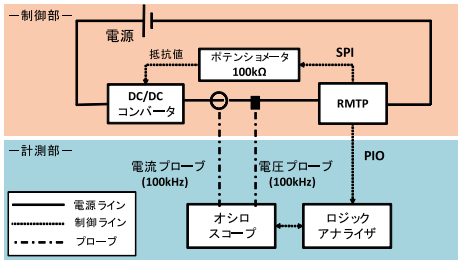
表 1　評価用のシステム
Table 1　System for evaluation.

α	f(MHz)	V_{dd}(Volt)	E(Ws)
0.25	7.81	0.83	0.17
0.33	10.41	0.86	0.21
0.50	15.63	0.93	0.29
1.00	31.25	1.05	0.48

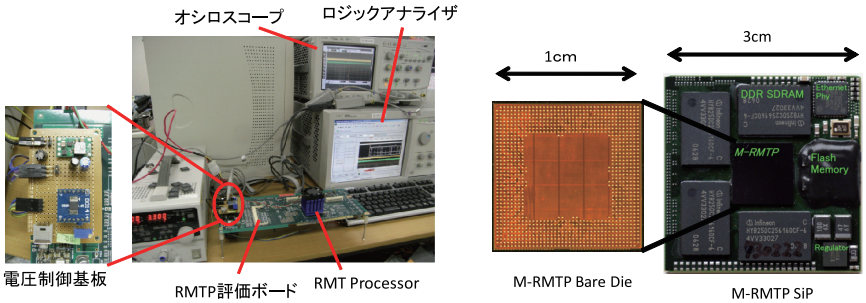
表 1 に本評価に使用した選択可能な α，動作周波数 f(MHz) とそれに対応する動作電圧 V_{dd}(Volt)，そして平均消費電力量 E(Ws) を示す．ある動作周波数に対する動作電圧は実験的に求めた．安定性は，四則演算するプログラムを一定時間流すことで確認した．動作周波数は，RMTP のシステムレジスタに値をセットすることで，次のクロックから変化する．動作電圧は，RMTP から外部のレギュレータへ信号を送ることで変化する．電圧を変更する場合，信号を送信してから実際に電圧が変わるまでに，約 100 ns の時間差が生じる．これは RMTP が 31.25 MHz で動作している場合，約 3 クロックに相当する．電圧設定後，十分な時間待つことによりこの時間差に対処する．

OS は我々の研究室で独自に開発している軽量 OS Favor を使用した．OS のスケジューラを実行するときには，スケジューラのオーバヘッドを最小にするために，動作周波数を最大にする．

図 5 (a) に評価環境の概要を，図 5 (b) に評価システムの外観を，図 5 (c) に RMTP の外観をそれぞれ示す．ロジックアナライザとオシロスコープは制御線で接続され同期がとられているとともに，RMTP から Parallel I/O（PIO）で制御されている．制御部では，ポテンショメータと DC/DC コンバータを用いて動作電圧の制御を行う．まず，RMTP は Serial Peripheral Interface（SPI）を用いて，ポテンショメータに指令値の信号を送り，ポテンショメータの抵抗値を変化させる．そのポテンショメータの抵抗値により，DC/DC コンバータは電圧を変化させる．計測部では，オシロスコープとロジックアナライザを用いて消費電力量を計測する．オシロスコープの電流プローブと電圧プローブを用いて電流と電



(a) 概要図



(b) 外観　　(c) RMTP

図 5　評価環境
Fig. 5　Evaluation enviroment.

圧を連続的に計測し，ロジックアナライザが測定データをファイルに保存する．RMTP は PIO を用いてロジックアナライザとオシロスコープを制御（計測の開始と終了の管理など）する．オシロスコープによって計測された一定間隔ごとの電流値と電圧値を掛け合わせ消費電力量を求める．

オシロスコープによって k 回目にサンプリングされた電流値と電圧値をそれぞれ A_k，V_k，総サンプリング回数を N とする．評価指標である $Energy$ を以下のように定義する．

$$Energy = \frac{1}{N}\sum_{k=1}^{N}(A_k \times V_k) \tag{1}$$

5.2　消費電力量に関する結果と考察

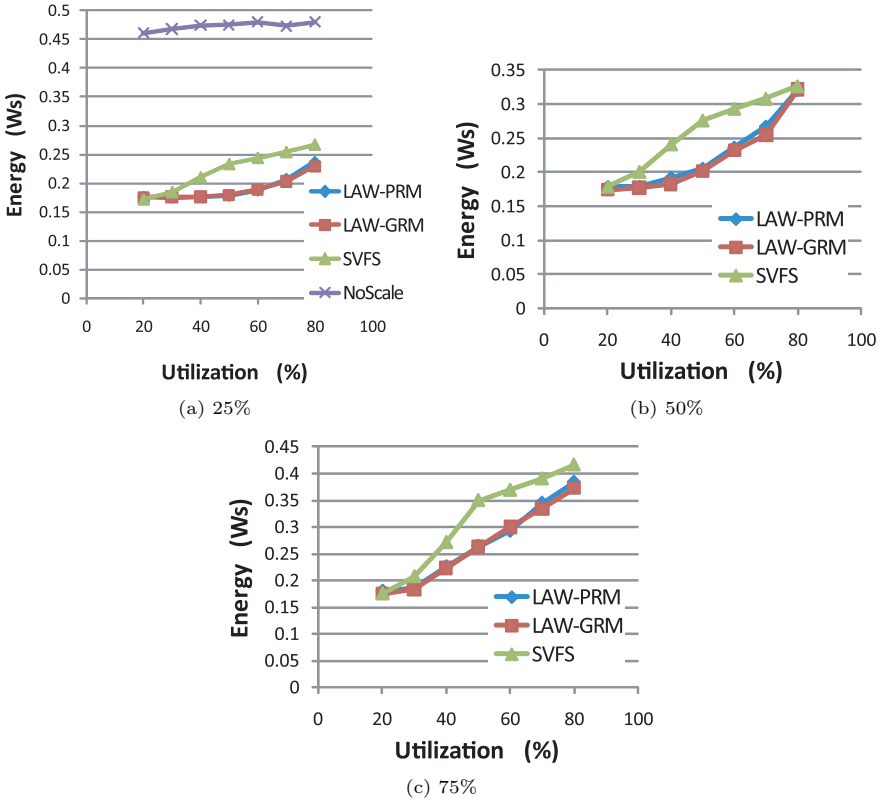
消費電力量の評価を以下のように行う．比較対象には Static Voltage and Frequency Scaling（SVFS）[5] を用いた．SVFS ではまず，システムの実行前にすべてのタスクがデッドラインを守ることができる最低動作周波数を計算する．そしてタスクを実行する際にはつねにこの周波数で実行する手法である．ただし，実行するタスクがなくなった場合には，システムが設定可能な最低動作周波数を設定する．SMT/CMP 向けの手法は Neils ら[14] や船岡ら[16] によって提案されているが，これらは動的優先度スケジューリング向けの手法である．我々の知る限り，固定優先度スケジューリング向けの手法は存在しなかったため，比較対象は SVFS のみとした．

表 2 評価用のタスク
Table 2　Tasks for evaluation.

Group	Tasks
A	$\tau(2C, 2T)$，$\tau(3C, 3T)$
B	$\tau(15C, 15T)$，$\tau(20C, 20T)$
C	$\tau(100C, 100T)$，$\tau(200C, 200T)$

表 2 に本評価で使用したタスクを示す．周期が短いタスクのグループを A，中間のグループを B，長いグループ C とし，この組合せによってタスクセットを構成する．これらのタスクは，時間粒度が異なる様々なタスクが存在するロボットの分散制御[17] を想定している．具体的には，グループ A はアクチュエータ/センサのローカル制御，グループ B はシステム全体のグローバル制御，グループ C はプランニングや地図生成などを想定している．T はタイマ割込みの周期であり，$T = 10$ [msec] である．C の具体的な数値は，タスクセットのプロセッサ使用率から計算される．以下，すべてのタスクがデッドラインを守っている範囲で評価を行う．タスクセットのスレッドあたりのプロセッサ使用率を [20%, 80%] の間に設定し，10%ごとにそれぞれのアルゴリズムで消費電力量を計測する．計測時間は [0, 600T] で，オシロスコープのサンプリング周期は 100 kHz である．LAW-PRM では，システムの実行前にタスクを各 LP に割り当てる必要があるが，本評価では周期の短いタスクから順に最も負荷の小さい LP にタスクを割り当てる．

図 6 に，スレッド数が 2 で，様々な種類の周期タスクで構成されるタスクセット（A+B+C）における結果を示す．(a)，(b)，(c) はそれぞれ，最悪実行時間（WCET）と実際の実行時間（ACET）の割合が 25%，50%，75%の場合の評価である．(a) においてのみ，動作電圧と動作周波数の制御を行わない場合（NoScale）との比較も行っている．図の横軸にタスクセットのスレッドあたりのプロセッサ使用率（%），縦軸に Energy（Ws）を示している．最悪実行時間と実際の実行時間の比によらず，提案手法は静的な手法である SVFS よ

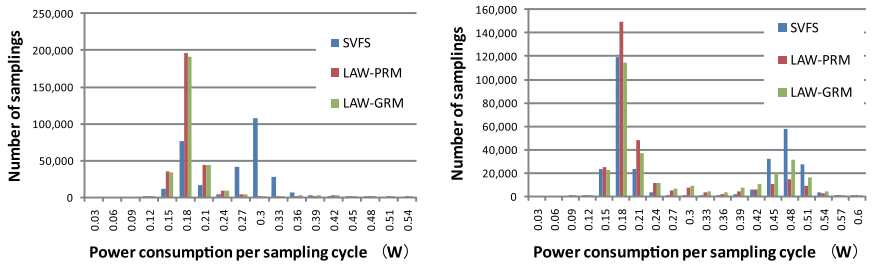


(a) 25%　(b) 50%

(c) 75%

図 6　各 WCET と ACET の比における平均消費電力量
（タスクセット A+B+C，m = 2）
Fig. 6　Average energy consumption with ratios of ACET divided by WCET (Taskset A+B+C, m = 2).

りも消費電力量を削減できていることが分かる．ただし，タスクセットのプロセッサ使用率が 30%より小さくなると，SVFS と提案手法の消費電力量はほとんど同じになる．これは，SVFS がシステムの実行前に決定した動作周波数が，システムが設定可能な最低動作周波数（$\alpha = 0.25$）よりも小さくなったためである．提案手法の SVFS に対する最大の消費電力量削減率は約 32%であった．また，提案手法の NoScale に対する最大の消費電力量削減率は

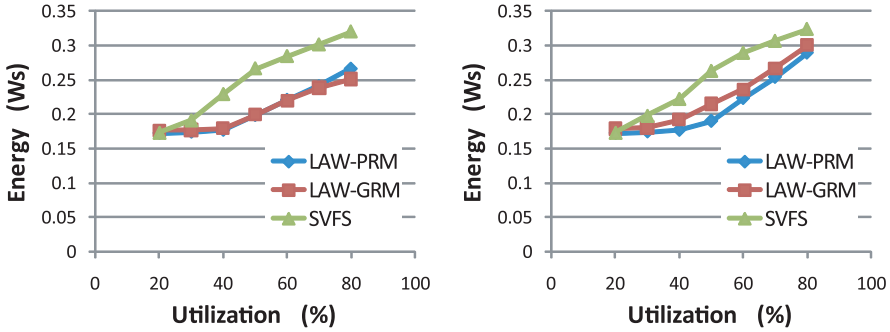


(a) Taskset A+B+C, Utilization 40%　(b) Taskset 2A+2B+2C, Utilization 60%

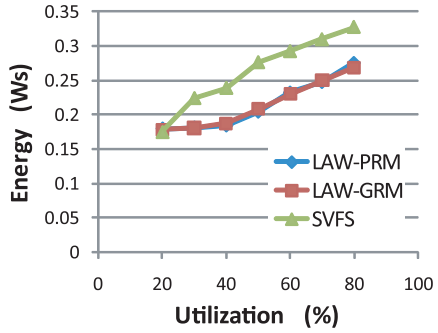
図 7　消費電力の分布（2 スレッド，ACET/WCET=50%）
Fig. 7　Distribution of Energy (2thread, ACET/WCET=50%).

約 63%であった．

図 7 に，サンプリング周期あたりの消費電力の分布を示す．横軸はサンプリング周期あたりの消費電力であり，縦軸はサンプリング数である．最悪実行時間と実際の実行時間の割合は 50%であり，スレッド数は 2 である．図 7 (a) はスレッドあたりのプロセッサ使用率が 40%であり，タスクセットが A+B+C の場合である．また図 7 (b) は，スレッドあたりのプロセッサ使用率が 60%であり，タスクセットが 2A+2B+2C の場合である．図 7 (a) より，LAW-PRM と LAW-GRM は高い動作周波数をほとんど選択していないことが分かる．提案手法はスケジューラが実行された時刻においてタスクセットがスケジューリング可能な最低動作周波数を算出する手法である．最悪実行時間と実際の実行時間が同じである場合には，他のタスクは高い動作周波数で実行しなければならない．しかしながら，実際の実行時間が最悪の実行時間よりも短いために，結果的に高い動作周波数で実行する必要がなくなったと考えられる．それに対して，SVFS は消費電力の分布のピークが 2 つ存在する．SVFS はシステムの実行前に，タスクセットがスケジューリング可能な最低動作周波数を算出し，タスクを実行する際はつねに算出した動作周波数で実行する手法である．実行するタスクが存在しない場合にはシステムで設定可能な最低動作周波数を設定する．そのため，システム実行前に算出した動作周波数における消費電力と，システムの最低動作周波数における消費電力の 2 カ所がピークとなったと考えられる．タスク数とスレッドあたりのプロセッサ使用率が図 7 (a) と比べ大きい図 7 (b) では，LAW-PRM と LAW-GRM でもピークが 2 つ存在する．しかしながら，提案手法は SVFS と比べると高い消費電力におけるサンプリング数が少ないことが分かる．そのため，提案手法が SVFS よりも消費電力量を削減でき



(a) A+C+C　(b) A+A+C



(c) A+B+B

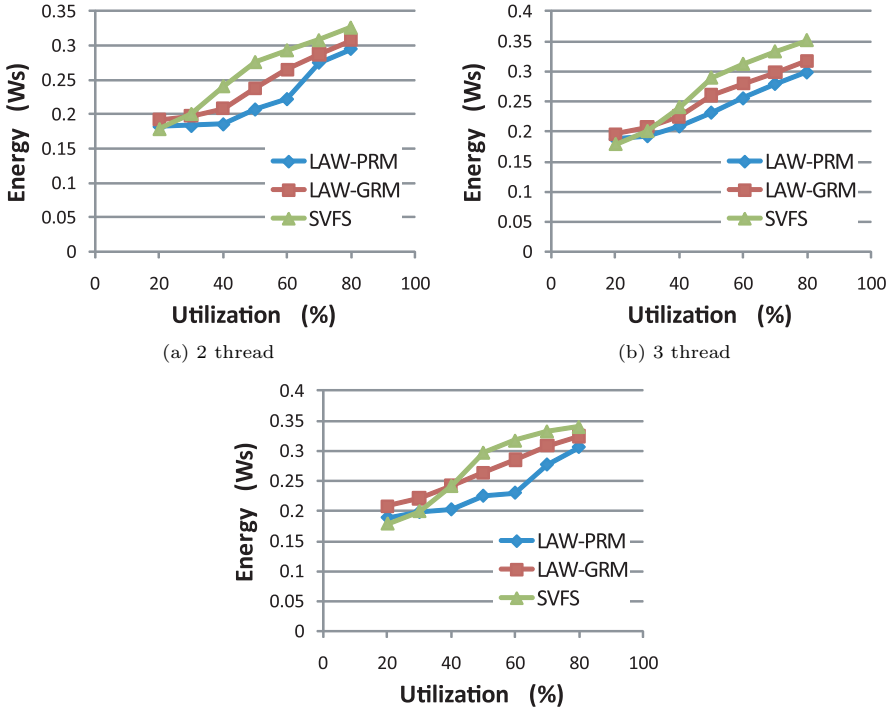
図 8　各タスクセットにおける平均消費電力量
(ACET/WCET = 50%, m = 2)
Fig. 8　Average energy consumption with tasksets (ACET/WCET = 50%, m = 2).

たと考えられる．

図 8 に，スレッド数が 2 で，最悪実行時間と実際の実行時間の割合が 50%の場合の結果を示す．(a)，(b)，(c) はそれぞれ，タスクセットが A+C+C，A+A+C，A+B+B の場合の評価である．タスクの組合せによらず，提案手法は SVFS よりも消費電力量を削減できていることが分かる．LAW-PRM と LAW-GRM を比べると，図 8 (b) の場合に LAW-PRM が LAW-GRM よりも消費電力量を削減していることが分かる．これは，スケジューラの実行時間の差によるものと考えられる．パーティショニング方式では各 LP ごとにレディキューがあるが，グローバルスケジューリング方式では 1 つのレディキューしかない．本評価で使用した Favor OS は，複数のスレッドが同時に同じキューへアクセスすることができない．短い周期のタスクが増えると，タスクがリリースされる回数が増加するため，タスクが終了する際に呼ばれるスケジューラの実行回数も増加する．そのため，グローバルスケジューリング方式では 1 つのキューへのアクセスが集中し，結果的にスケジューラの実行時間が増加してしまう．前述のように，スケジューラの実行時には動作周波数を最大にするため，グローバルスケジューリング方式向けの LAW-GRM の消費電力量が増加したと考えられる．

図 9 に，最悪実行時間と実際の実行時間の割合が 50%で，タスクセットが 2A+2B+2C の場合の結果を示す．(a)，(b)，(c) はそれぞれ，スレッド数が 2，3，4 の場合の評価である．図 8 (b) と図 9 (a) を比べると，図 9 (a) の方が LAW-PRM と LAW-GRM の消費電力量の差が大きいことが分かる．つまり，タスク数が増加すると，LAW-GRM の消費電力の削減量が低下する．本論文の提案手法は，ウィンドウ内でリリースされた高優先度タスクはウィンドウ内ですべて実行されると仮定している．しかしながら，実際にはウィンドウ外で実行される場合があり，高優先度タスクにブロックされる時間の予測値と実際にブロックされる時間の間に誤差が生じる．タスク数が多くなると，この誤差は大きくなる．LAW-PRM は，同じ LP に割り当てられたタスクから高優先度タスクにブロックされる時間を計算するのに対し，LAW-GRM はすべてのタスクからブロックされる時間を計算するため，LAW-PRM よりも誤差が大きくなる．誤差が大きくなると，余裕時間を少なく見積もってしまい，その結果として選択する動作周波数が高くなり，消費電力量を十分に削減できなくなると考えられる．図 7 (b) では，LAW-GRM が LAW-PRM に比べ高い動作周波数をより多く選択していることが分かる．高い動作周波数を多く選択してしまった結果，LAW-GRM は LAW-PRM に比べ消費電力の削減量が低下してしまったと考えられる．

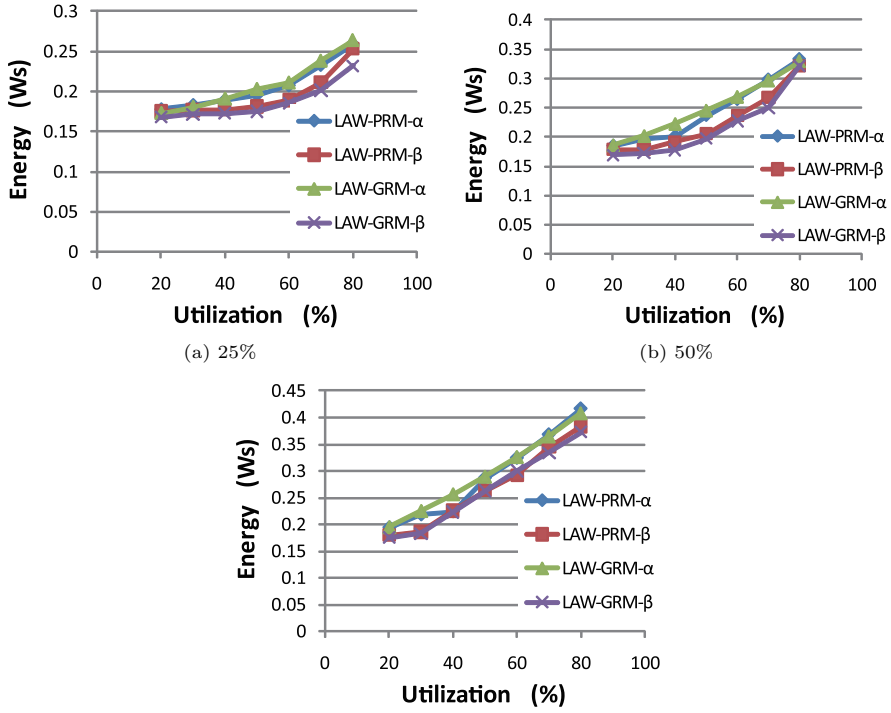
次に，スレッド数が変化したときの消費電力量の違いに着目する．図 9 (b) において LAW-PRM の SVFS に対する消費電力量の削減量が，図 9 (a) および図 9 (c) での削減量と比べ少ない．これは，各スレッドへのタスクの割当てが影響していると考えられる．図 9 で使用したタスクセットは，周期が短いタスク，中間のタスク，長いタスクがそれぞれ 4 つずつで構成されている．スレッド数が 3 の場合，各スレッドに割り当てられるタスクの周期の長さが均等でなくなる．LAW-PRM は，それぞれの LP ごとに計算された動作周波数の最大値をシステムの動作周波数に決定するアルゴリズムである．したがって，高い動作周波数で実行しなければならない LP が 1 つでもあれば，他の LP で実行するタスクが存在しなくても高い動作周波数で実行しなければならない．そのため，タスクの周期に偏りがあると消費



(a) 2 thread　(b) 3 thread

(c) 4 thread

図 9　各 thread 数における平均消費電力量
（ACET/WCET = 50%，タスクセット 2A+2B+2C）
Fig. 9　Average energy consumption with 2, 3, and 4 threads (ACET/WCET = 50%, Taskset 2A+2B+2C).



(a) 25%　(b) 50%

(c) 75%

図 10　手法 α と手法 β の平均消費電力量の比較結果
（2 スレッド，タスクセット A+B+C）
Fig. 10　The result of comparison of energy consumption on method-α with method-β (2 threads, Taskset A+B+C).

電力の削減量が低下すると考えられる．

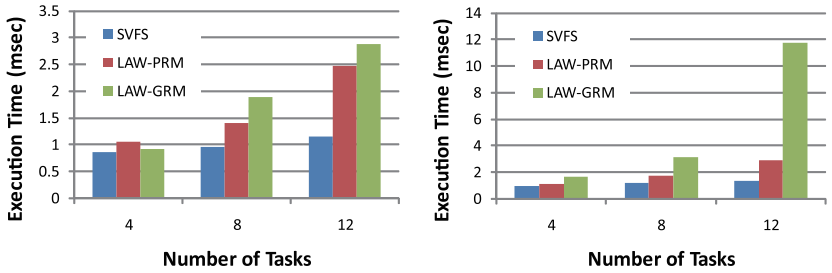
5.3　余裕時間の分配方法に関する比較評価と考察

本章では，提案手法のアルゴリズムにより算出した余裕時間を全タスクで平均的に分配する手法（手法 α）と，カレントタスクのみに分配する手法（手法 β）の消費電力量を比較する．

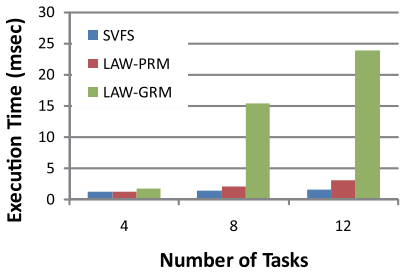
図 10 に，手法 α と手法 β を比較した結果を示す．スレッド数は 2 であり，タスクセットは A+B+C である．(a)，(b)，(c) はそれぞれ，最悪実行時間と実際の実行時間の割合が 25%，50%，75%の場合の評価である．最悪実行時間と実際の実行時間の比がいずれの場合においても，手法 β が手法 α よりも消費電力量を削減できていることが分かる．この比較結果から，最悪実行時間と実際の実行時間の差がある場合には，余裕時間を分け合った手法よりもカレントタスクのみに利用した本研究の手法が消費電力量を削減できることが分かる．

5.4　スケジューラの実行時間に関する評価と考察

図 11 に，提案アルゴリズムの実行を含むスケジューラの実行時間の最悪値を示す．タスクはすべて同じ周期（100 msec）で，タスクは何も実行せずに終了する．提案アルゴリズム



(a) 2 thread　(b) 3 thread



(c) 4 thread

図 11　各 thread 数におけるスケジューラの実行時間の最悪値
Fig. 11　Worst case execution time of scheduler with 2, 3 and 4 threads.

は動作周波数の計算は行うが，実際にシステムには設定せず動作周波数は変化させない．タスクを 20 周期実行し，最悪値を計測する．3 スレッド以上になると，LAW-GRM の結果が，LAW-PRM および SVFS の結果に比べて大幅に増加してしまっている．スケジューラを実行する場合，タスクキューにアクセスすることができるのは同時に 1 スレッドだけである．パーティションスケジュールの場合はタスクキューはスレッドごとにあるため，タスクキューへのアクセス競合は起こらない．それに対してグローバルスケジューリングでは，タスクキューが 1 つしかないため，アクセスの際に競合が起こってしまう．複数スレッドがほぼ同時に実行を終了した場合，最後にキューにアクセスするスレッドは他のスレッドがタスクキューにアクセスしている間待つことになる．そのため，スレッド数が増加すると，LAW-GRM のスケジューラの実行時間が大幅に増加したと考えられる．

6. 結　　論

本論文では，SMT/CMP 向け固定優先度スケジューリング用電圧周波数制御アルゴリズムとして LAW on Partition-RM（LAW-PRM）と LAW on Global-RM（LAW-GRM）を提案し，その有効性を実機評価した．RMTP を用いた評価では，電圧周波数制御を行わない場合と比べ最大で 63%，静的に動作周波数を決定する手法（SVFS）と比べ最大で 32%消費電力量を削減できた．

謝辞　本研究は科学技術新興機構 CREST によるものであり，一部は文部科学省グローバル COE プログラム「環境共生・安全システムデザインの先導拠点」によるものであることを記し，謝意を表す．

参 考 文 献

1) Liu, C.L. and Layland, J.W.: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, *J. ACM*, Vol.20, No.1, pp.46–61 (1973).
2) Burd, T.D. and Brodersen, R.W.: Energy Efficient CMOS Microprocessor Design, *Proc. 28th Annual Hawaii International Conference on System Sciences*, pp.288–297 (1995).
3) Tullsen, D.M., Eggers, S.J. and Levy, H.M.: Simultaneous Multithreading: Maximizing On-Chip Parallelism, *Proc. 22nd Annual International Symposium on Computer Architecture*, pp.392–403 (1995).
4) Olukotun, K., Nayfe, B., Hammond, L., Wilson, K. and Chang, K.: The Case for a Single-Chip Multiprocessor, *Proc. International Conference on Architectural Support for Programming Languages and Operating Systems*, pp.2–11 (1996).
5) Pillai, P. and Shin, K.G.: Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems, *Proc. ACM Symposium on Operating Systems Principles*, pp.89–102 (2001).
6) 池田雄児，加藤真平，山﨑信行：固定優先度スケジューリング向け実時間電圧周波数制御，*SACSIS*, pp.407–414 (2009).
7) Chen, J.J. and Kuo, T.W.: Procrastination Determination for Periodic Real-Time Tasks in Leakage-Aware Dynamic Voltage Scaling Systems, *Proc. Int'l Conf. Computer-Aided Design*, pp.289–294 (2007).
8) Chen, J.J., Hsu, H.R., Chuang, K.H., Yang, C.L., Pang, A.C. and Kuo, T.W.: Multiprocessor energy-efficient scheduling with task migration considerations, *Proc. 16th Euromicro Conference on Real-Time Systems*, pp.101–108 (2004).
9) Chen, J.J., Hsu, H.R. and Kuo, T.W.: Leakage-aware energy-efficient scheduling

of real-time tasks in multiprocessor systems, *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium* (*RTAS*), pp.408–417 (2006).
10) Shao, Z., Wang, M., Chen, Y., Xue, C., Qiu, M., Yang, L.T. and Sha, E.H.-M.: Real-Time Dynamic Voltage Loop Scheduling for Multi-Core Embedded Systems, *IEEE Trans. Circuits and Systems II* (*TCAS-II*), Vol.54, No.5, pp.445–449 (2007).
11) Zhu, D., Melhem, R. and Childers, B.R.: Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multiprocessor Real-Time Systems, *IEEE Trans. Parallel and Distributed Systems* (*TPDS*), Vol.14, No.7, pp.686–700 (2003).
12) Yamasaki, N.: Responsive Multithreaded Processor for Distributed Real-Time Systems, *Journal of Robotics and Mechatronics*, pp.130–141 (2005).
13) Yang, C.Y., Chen, J.J., Hsu, H.R. and Kuo, T.W.: An approximation algorithm for energy-efficient scheduling on a chip multiprocessor, *Proc. 8th Conference of Design, Automation, and Test in Europe* (*DATE*), pp.468–473 (2005).
14) Neils, V., Goossens, J., Devillers, R. and Milojevic, D.: Power-Aware Real-Time Scheduling upon Identical Multiprocessor Platforms, *IEEE International Conference on Sensor Networks Ubiquitous and Trusworthy Computing*, pp.209–216 (2008).
15) Tullsen, D.M., Eggers, S.J. and Levy, H.M.: Simultaneous Multithreading: Maximizing On-Chip Parallelism, *Proc. 22nd Annual International Symposium on Computer Architecture*, pp.392–403 (1995).
16) 船岡健司，加藤真平，山﨑信行：マルチプロセッサ用の実時間電圧周波数制御，情報処理学会論文誌，Vol.1, No.2, pp.96–110 (2008).
17) Matsui, T., Hirukawa, H., Yamasaki, N., Ishikawa, H., Kagami, S., Kanehiro, F., Saito, H. and Inamura, T.: Distributed Real-Time Processing for Humanoid Robots, *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp.205–210 (2005).

(平成 22 年 3 月 1 日受付)
(平成 22 年 9 月 17 日採録)

池田　雄児（正会員）
1986 年生．2009 年慶應義塾大学理工学部情報工学科卒業．現在，同大学大学院理工学研究科開放環境科学専攻修士課程在籍．省電力リアルタイムシステム，オペレーティングシステム等の研究に従事．

加藤　真平（正会員）
1982 年生．2004 年慶應義塾大学理工学部情報工学科卒業．2008 年同大学大学院理工学研究科開放環境科学専攻博士課程修了．博士（工学）．2009 年度 4 月より東京大学大学院情報理工学系研究科特別研究員．現在，米国カーネギーメロン大学訪問研究員としてオペレーティングシステム，リアルタイムシステム，高性能システム等の研究に従事．

山﨑　信行（正会員）
1991 年慶應義塾大学理工学部物理学科卒業．1996 年同大学大学院理工学研究科計算機科学専攻博士課程修了．博士（工学）．同年電子技術総合研究所入所．1998 年 10 月慶應義塾大学理工学部情報工学科助手．同専任講師を経て 2004 年 4 月より同助教授（現，准教授）．リアルタイムシステム，プロセッサアーキテクチャ，並列分散処理，システム LSI，ロボティクス等の研究に従事．日本ロボット学会，IEEE 各会員．